Security Framework based on Fault Tolerance in Hand-held Devices

Research proposal submitted for the Pre-submission Defense towards partial fulfillment of the Degree of Doctor of Philosophy (Ph. D)

Atanu Datta

Ph D scholar (Registration no: SVU/PhD/SCI/CS/001/2021)

School of Computer Science, Swami Vivekananda University, Barrackpore, West Bengal

Under the Guidance of

Prof. Somsubhra Gupta (Supervisor)

Swami Vivekananda University, Barrackpore, West Bengal

Prof. Subhranil Som (Jt. Supervisor)

Bhairab Ganguly College, Kolkata, West Bengal

Swami Vivekananda University

BARRACKPORE, WEST BENGAL – 700121.

Abstract: Mobile data security refers to the exchange of wireless information and services electronically with citizens' businesses between banks using information and communication technology. This synopsis imposes a security mechanism of data in the banking and financial sector with the help of AES algorithm.

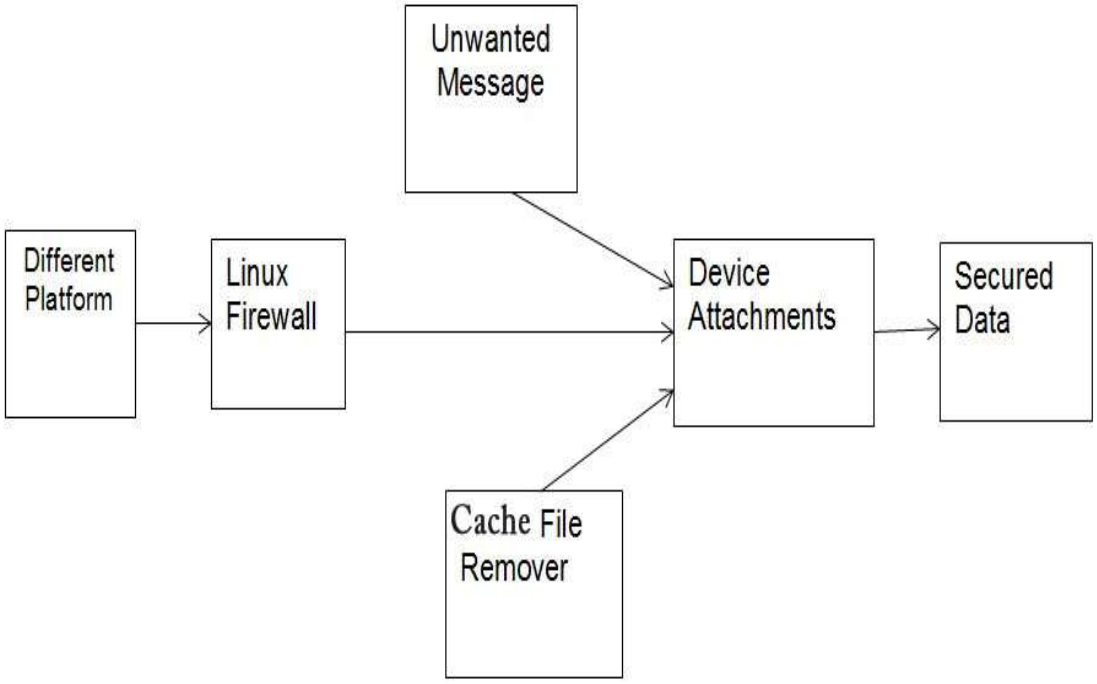
During the transaction of data and information, the hackers can listen to, modify or steal information in different ways. This synopsis proposes to find out a suitable encryption technique to prevent attacks caused by hackers in a financial transaction. In this synopsis, we have encapsulated algorithm in Object-Oriented Model with SIGMA LFSR in pipeline and for implementation of confidentially of the message to improve the citizens' confidence on online banking transaction especially on client-side application-based mobile handheld devices.

As the proliferation of mobile phones continues in daily life, it significantly impacts various aspects, notably mobile phone security. In many cases, crucial credentials are stored on these devices. Smartphones harbor a plethora of sensitive data that require safeguarding. This synopsis elucidates on employing cryptographic techniques to fortify the security of your mobile or smartphone.

Mobile banking is fast and convenient, but it faces security challenges. Many organizations now include mobile Payment system as a key part of their growth strategy, and its use is growing. Security issues in mobile Payment system have been extensively researched. This review analysed, revealing various challenges. Security concerns are a common reason why many consumers hesitate to adopt mobile banking. This review aims to address these concerns and help financial institutions implement mobile banking successfully. Mobile Payment system is emerging as a crucial channel in the financial sector, but user confidence in security is a major obstacle. Understanding the market and ecosystem is vital to tackle these security challenges and mitigate new risks associated with mobile Payment system.

Smartphones have now become a vital tool in many people's everyday life, even to the level of addiction for some. Cell phones are a critical piece of our regular use with the increasing of smartphone, the number of risk of data in smartphone is also increased. This synopsis provides regarding smartphone security and their vulnerabilities and how to be secure them and data security challenges and awareness about them. Data security on cell phones is a new and developing worry, especially when the data is expensive or sensitive. Despite the fact that there are numerous data protection solutions for cell phones, no research has been conducted. Users' expectations for such solutions We use a different strategy in this study. User-centric approach to the data protection challenge on cell phones, and examine the needs for data protection systems from the perspective of users' viewpoints. We elicit the types of data that users want to keep private. investigate current users' data-protection habits, and demonstrate how the security requirements for various data kinds varies.

Research Road Map



Chapter Description

Chapter1: Introduction

Chapter 2: Literature Survey

Chapter 3: Methodology Aspects: Security and Firewall

- 3.1 Encryption Key
- 3.2 Pipelining SIGMA LFSR model
- 3.3 Proposed Design

Chapter 4: Methodology Aspects: Protection of unwanted Messages

- 4.1 Security Challenges in Mobile Device Operating Systems
- 4.2 Network and Transport Frequency Challenges:

Chapter 5: Methodology Aspects: Cleaning of Cache

- 5.1 Dynamic Cache Cleaner Model
- 5.2 Memory Consumption

Chapter 6: Methodology Aspects: Device Attachment and Data Security

- 6.1 Vulnerabilities in USB Connection
- 6.2 Scenarios
- 6.3 Implementation

Chapter 7: Result Analysis and Conclusion

Chapter 1: Introduction

Financial data transfer (from Banking, E-wallet, online purchase, or in a certain context government tax payment)

Refers to the utilization of information technology for exchanging information and services with

citizens, businesses, and other entities within financial institutions Currently, the authentication of electronic documents and their efficient packaging stands as the primary constraint for the development of financial transaction applications. There are additional unresolved issues such as ensuring protection, validity, uniqueness, and traceability of e-documents, preventing illegal duplication, and ensuring tamper-proofing.

The Indian Government's National E-Transaction Plan aims to establish the groundwork and foster the long-term expansion of e-transactions within and beyond the nation, particularly for Indian financial institutions. Some crucial client-side applications in the financial sector include Income Tax filing, Banking services, provident fund status, passport and visa information, voter ID/National citizen card status, Trade license keys or agreements, PAN, Aadhar card verification, among others. Additionally, attackers may manipulate base information as per their needs. Therefore, to ensure swift communication and mitigate unauthorized access in information and communication technologies (ICT), employing secure cryptographic approaches while transferring data over the internet is imperative.

The cryptographic approach can be categorized into three types: asymmetric cryptosystem, symmetric cryptosystem, and digital signatures. In a symmetric cryptosystem, both the sender and receiver utilize the same key, ensuring confidentiality and privacy. On the other hand, an asymmetric cryptosystem employs different keys for encryption and decryption, primarily facilitating key exchange and authentication. Digital signatures encrypt data irreversibly, ensuring data integrity.

Encryption, a novel technology, safeguards data by scrambling it and remains unique to their respective key sizes, providing robust protection. Numerous algorithms are involved in this process, such as Triple DES (Data Encryption Standard), RSA, and AES (Advanced Encryption Standard). Modern cryptography functions include DES and AES under symmetric cryptosystem, and RSA under asymmetric cryptosystem.

Cryptography is the art of employing mathematical principles to encode and decode data. It empowers you to secure sensitive information by either storing it securely or transmitting it across vulnerable networks, such as the internet, ensuring that only the intended recipient can decipher it. The aim of online financial transactions via mobile devices is to facilitate seamless interaction among various components of the online transaction process. To achieve this, we employ an asymmetric cryptosystem.

The study finds that consumers consider handset compatibility, security, and service standardization as critical issues in mobile banking. These results offer practical recommendations for addressing security challenges in mobile Payment system. This synopsis reports the findings of a systematic review of security challenges in mobile banking payments. It gives an overview of the current state of these challenges and their implications. The research also reveals interconnected challenges in mobile banking payments and smartphone security. These findings emphasize the need for more research in this area. Efforts like workshops, conferences, and projects are being established to address these challenges in online payments.

Chapter 2: Literature Review

I studied different security-related challenges for mobile users, mobile threats, mobile vulnerabilities. Different types of mobile risks involved in their study are physical based threats, application-based threats, network-based threats and Web-based threats. A botnet is one of earnest money related threat in all mobile vulnerabilities. According to them one of important security defence mechanism for data privacy and mobile security is Biometric authentication. Security mechanisms need to be involved in every stage of mobile application development. Their results show that apps with remote monitoring functionalities have the greatest number of vulnerabilities and also vulnerabilities in these apps contain

high-risk levels. Classified cryptographic weakness in mobile applications into weak cryptographic algorithms, weak cryptographic keys, weak implementation of cryptographic algorithms and weak parameters. They downloaded 49 random android apps from Google play store, and they performed manual static analysis and dynamic analysis of those applications. Their results show that almost 87.8% of Android apps are using weak cryptographic algorithms and 12.2% of android apps not implemented any cryptographic algorithm at all. The proposed protocol based on commitment scheme and it will stop communications if it encounters authentication failure. They stated that because of binding/hiding nature of protocol it is very effective to prevent Man in the middle attacks in wireless communications. stated that improvements in new technologies and developments in security measures needed to be parallel. According to Routine activity theory crime occurs when there is a weak guardianship, targeted device and motivated attacker. Cloud storage apps like DropBox, Google Drive, One Drive, etc., are favourite targets for attackers because of their functionality to store a significant amount of user data. They investigated on compromised celebrities iCloud accounts and stated that most of the attacks are targeted attacks on security questions, usernames, and passwords. stated that there is no complete solution to prevent mobile security problems. The main issues with the mobile security are implementing proper security policies, integrating current security and protecting data in mobile devices. To secure business documents and data, corporates need to implement a secure environment for mobile devices, threat management and security policies need to be independent of devices and operating system used in them.

Gap Analysis:

This research focuses on enhancing the secure development of financial transactions in mobile environments. In today's world, smartphones have become central to individuals' daily activities, serving as the primary device for most interactions. However, this increased reliance on mobile devices has led to a rise in security threats compared to traditional online platforms. For instance, users often download applications based on descriptions or reviews without thoroughly evaluating their safety. These apps, as executable files, can be exploited for malicious purposes due to the generally weak implementation of information security measures on smartphones. Additionally, the regulatory framework governing mobile privacy tends to be less robust than that for traditional platforms. On average, users have over 100 apps on their smartphones, some of which may pose security risks. Many apps allow developers to collect users' personal data for various purposes, often without users' knowledge or consent. The rapid proliferation of smartphones has turned the mobile ecosystem into a lucrative target for cybercriminals.

This research aims to address these vulnerabilities by providing protection for both external threats and internal risks posed by apps installed for other purposes. While existing studies primarily focus on safeguarding against external attacks, this work seeks to extend protection to internal sources, ensuring comprehensive security for mobile users.

Chapter 3: Methodology Aspects: Security and Firewall

- The Advanced Encryption Standard (AES) is a symmetric encryption technique that employs block encryption, dividing plaintext into blocks of 16 bytes each. It supports key lengths of 128, 192, and 256 bits. In this implementation, AES with a 128-bit key is utilized.
- The Data Encryption Standard (DES) is another symmetric encryption method that uses 64-bit block encryption with a 56-bit key length. Due to its relatively low security, DES is employed only as the first layer of encryption in this algorithm.
- AES is significantly faster than DES, often achieving speeds up to six times greater. While DES was once considered highly secure, its shorter key length made it vulnerable to key search attacks as computing power advanced. Although 3DES was developed to address these weaknesses, it was later deemed too slow for practical use.

Features of AES

- Utilizes symmetric keys and operates as a symmetric block cipher.
- Processes 128-bit blocks of data with key sizes of 128, 192, or 256 bits.
- Offers higher speed and greater security compared to 3DES.

The following is a representation of the algorithm:
Enter ——> 3DES Encryption ——> AES Encryption ——> Output
3DES decryption ——> AES decryption ——> output

The following is the steps for the Algorithm:

Step 1: The 3DES encryption formula:

1 C = E (K, P) / / C is the ciphertext, K is the 56-bit key, and P is the plaintext
2 The AES encryption formula:
3 C = E (K, P) / / C is ciphertext, K is 128-bit key, P is plaintext or ciphertext of the previous layer The DES decryption formula:
4 P = D (K, C) / / P is the plaintext, K is the 56-bit key, and C is the ciphertext The AES decryption formula:
5 P = D (K, C) / / P is plaintext, K is 128-bit key, C is ciphertext or plaintext of the previous layer

3.1 The Encryption Key and Its Expansion

Assuming a 128-bit key, the key is also arranged in the form of a matrix of 4 × 4 bytes. As with the input block, the first word from the key fills the first column of the matrix, and so on.

The four-column words of the key matrix are expanded into a schedule of 44 words. (As to how exactly this is done, we will explain that later.) Each round consumes four words from the key schedule. The figure below depicts the arrangement of the encryption key in the form of 4-byte words and the expansion of the key into a key schedule consisting of 44 4-byte words.

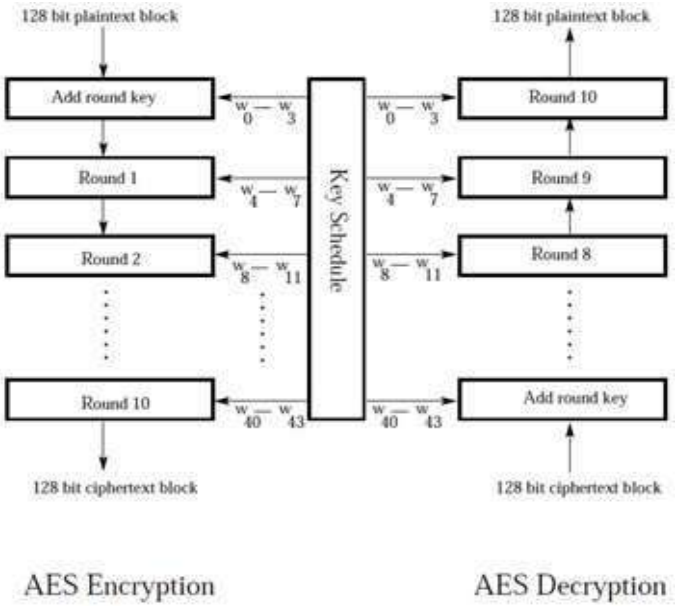


Figure 1: **Rounds of AES**

The encryption algorithm combining AES and DES offers a high level of security. Firstly, the DES algorithm uses a 56-bit key. While it may not be the most secure encryption method today, it still provides a certain degree of security in practical applications. Secondly, the AES algorithm supports key lengths of 128 bits and 256 bits, offering significantly higher security. Consequently, the combination of AES and DES algorithms can effectively resist brute-force attacks.

3.1.1 PORPOSED DESIGN

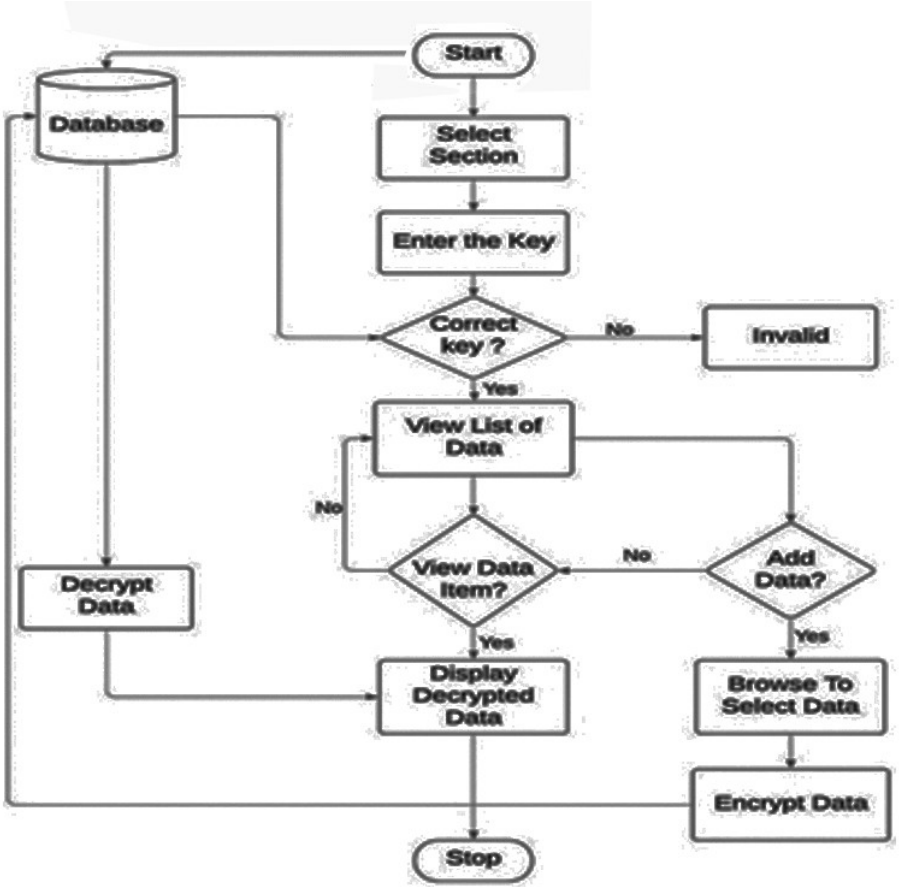


Figure-2: Proposed Flow Chart

3.2 Pipelining SIGMA LFSR model

Next the control will be redirected from AES to a SIGMA -LFSR channel through a pipeline, in which the unauthentic infiltration or invasion accessibility will be blocked. This means, the systems file and data has to be protected through this SIGMA-LFSR module through proper authentication scheme. The idea here is to provide hassle free accessibility to the authentic user at the same time generating a protective shield to the system jobs/files/ data through this SIGMA LFSR module from the infiltrators/ intruders.

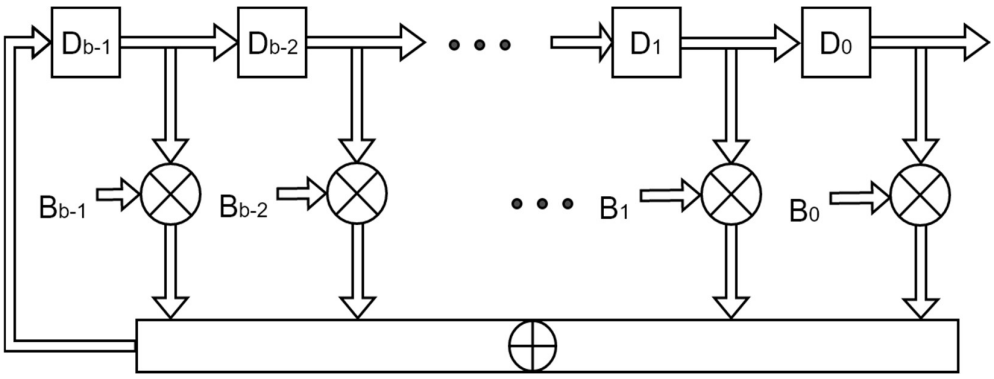


Figure. 3 : Block Diagram SIGMA-LFSR

Chapter 4: Methodology Aspects: Protection of unwanted Messages

The SMS-labelled message dataset used in this study was sourced from the Machine Learning repository. This dataset contains 5,559 labelled messages, with 4,812 classified as legitimate (ham) and 747 as spam, as summarized in Table 1. The dataset is organized into two main columns: one containing the message labels (ham or spam) and the other comprising the text messages. Additionally, it includes three unnamed columns. As highlighted in Table 1, 86.56% of the messages are ham, while 13.43% are spam.

Table 1: Type of features of dataset

Data Set	Legitimate	Spam	Total
SMS spam	3358	396	3754
DIT SMS spam	904	263	1167
British English SMS	550	88	638
Total	4,812 (86.56%)	747(13.43%)	5559

To ensure its suitability for machine learning applications, the dataset underwent preprocessing to address unnecessary features and refine the raw data. Preprocessing involves steps such as removing background noise, sampling, and formatting. This study utilizes a combination of content-based and user-based features to create a robust system for effectively detecting and classifying spam messages. Content-based features include words, phrases, and patterns commonly found in spam messages, while user-based features incorporate details like the sender's phone number, message frequency, and the time messages are sent. The SMS spam filtering system design includes several key components: data preprocessing, feature extraction, detection, and classification. Feature extraction transforms SMS messages into a structured set of features that can be processed by machine learning algorithms. For classification, this research employs the Bayes Theorem to differentiate between spam and ham messages.

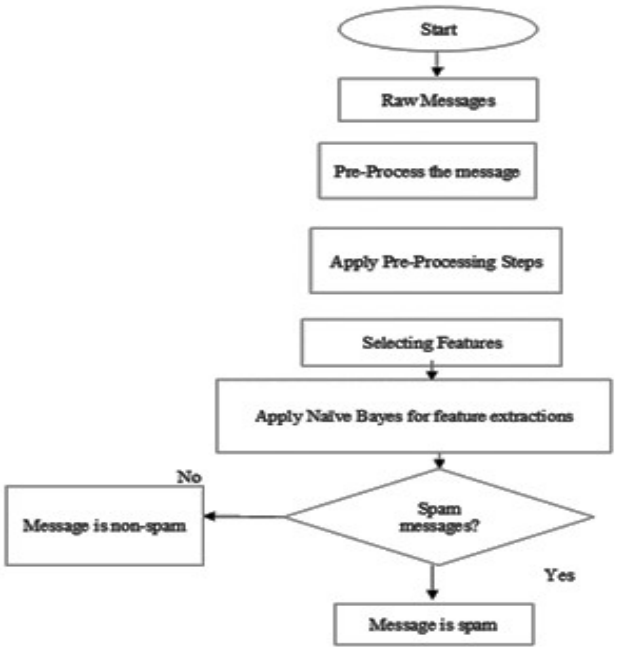


Figure 4: Flowchart of Spam filtering

Figure 4 illustrates the system's flow, beginning with raw data components, proceeding through preprocessing stages, and applying algorithmic steps to identify spam or non-spam messages on mobile devices. The system was implemented using Python and libraries such as scikit-learn and NLTK for preprocessing, feature extraction, and classification.

The Naïve Bayes classifier achieved an accuracy of over 97% in correctly classifying SMS messages as spam or non-spam. Figure 4 showcases the SMS spam filtering system's search feature, which allows users to search messages and evaluate their classification as spam or legitimate. This functionality enables efficient detection and classification of spam messages

Experiments and Result Analysis

- The formula for Bayes' theorem is given as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- Where,
- P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.
- P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.
- P(A) is Prior Probability: Probability of hypothesis before observing the evidence.
- P(B) is Marginal Probability: Probability of Evidence.

A short user feedback report is listed here. I have shared this application to my students' friends and native circle. After using this application, I request them to send a feedback report through google sheet. The report is listed below of Spam application users.

Table 2: General Feedback of users segregate by age.

	Worst (<40%)	Average (<70%)	Best (>70%)
Male Age 18-35	17.6	68.2	14.2
Female Age 18-35	15.5	73.3	11.2
Male Age 36-50	20.7	69.5	9.8
Female Age 36-50	18.6	65.8	13.6
Male Age Above 50	6.9	57.9	35.2
Female Age Above 50	10.7	63.2	26.1

By observing this feedback report I have found that the worst result come from Age group 31 to 50. ~~Its may be that~~ There is a possibility that they are using or registered various types of business houses so they are receiving various types of messages with various pattern. In this regard I am working for better result.

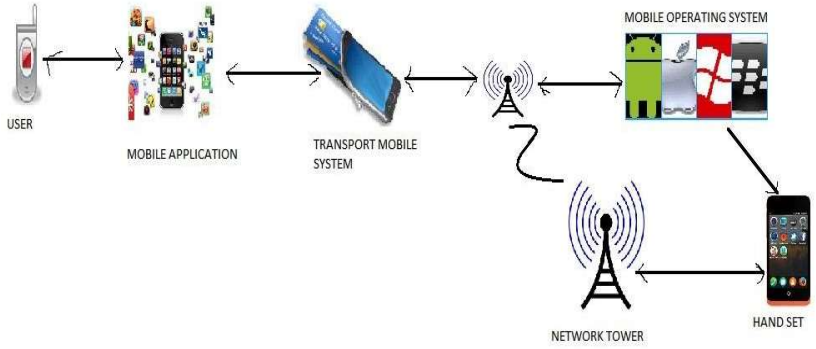


Figure -5. Network and Transport System

Network Challenges Mobile systems commonly use GSM and CDMA networks. Weaknesses in these networks pose challenges to mobile banking. Attacks on these systems can disrupt mobile network functionality, allowing attackers direct access to the mobile channel system.

GSM*:* GSM encryption has recently been cracked, with researchers and organizations discovering weaknesses in the A5/1 and A5/2 encryption algorithms, allowing for traffic interception and decryption. There are both active and passive tactics, with passive ones being more difficult to detect because they avoid transmitting more traffic.

CDMA: CDMA transmissions are more challenging to crack than GSM. CDMA uses a code and multiplexing technique, allowing it to handle more users on fewer cellular network towers.

Transport Challenges: The mobile transport system encompasses SMS, WAP, HTTP, TCP/IP, OTA, BLUETOOTH, and USSD, and how data is sent presents challenges to mobile banking payment systems.

HTTP and WAP Browser: Smartphones can use the HTTP protocol to access the internet, relying on communication protocols to enhance security, including SSL encryption. Wireless Application Protocol (WAP) is also used in mobile systems, allowing internet access via mobile devices.

TCP/IP and SMS: TCP/IP is used by most smartphones for internet communication, but it can be vulnerable to attacks on the IP layer, such as Routing Information Protocol attacks.

BLUETOOTH, OTA, and USSD: Bluetooth protocols can be exploited by attackers, resulting in issues like bluejacking and bluesnarfing. To address these concerns, many mobile handsets now disable Bluetooth by default. Over-the-Air (OTA) programming allows administrators to upgrade systems remotely.

Chapter 5: Methodology Aspects: Cleaning of Cache

Cache serves as a tool to store data transparently, facilitating quicker access for future requests. Subsequent data requests are swiftly met by retrieving information from the solicitation cache split up, obviating the necessity to establish network connections or access distant servers to satisfy the entreaty. fleeting housing data in the local cache significantly enhances the attainment of Android applications, preventing the unnecessary retrieval of recently accessed information.

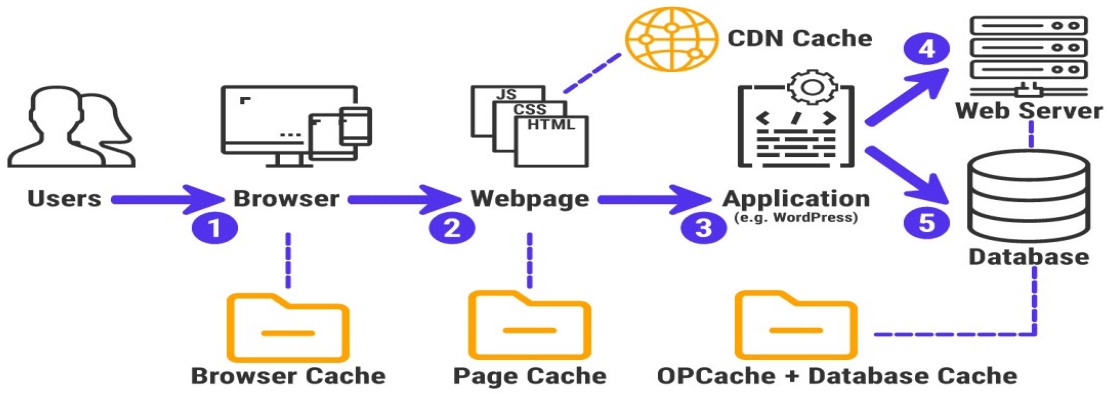


Figure-6: Cleaning of Cache Control Flow

Users may find that many of their applications exceed this recommended cache size when checking their Android mobile device's storage allocation for cached files. Notably, certain resource-intensive applications like web browsers are anticipated to exceed the recommended cache storage limit because of their frequent usage and versatility.

Catch Cleaning T1, T2 takes 20 minutes interval shows percentage of memory usage by Machine MC1, MC2, MC3, MC4.

Table-3: Cache Cleaning time comparison

	T1	T2	T3		T11	T12	T13
MC1	20	40	49	----	52	53	52
MC2	20	45	55	-----	55	54	55
MC3	18	48	51	-----	57	57	55
MC4	20	45	52	-----	52	51	50
-----	-----	-----	----	----	-----	----	-----

The constitutional storage measurement of mobile devices, although continuously expanding, still pales in comparison to that of desktops or laptops. Consequently, internal storage is considered valuable and limited. Additionally, as cached files accumulate and occupy the already limited space within the device, it negatively affects the device's overall performance.

5.1 Dynamic Cache Cleaner Model

The concept behind a dynamic cache cleaner is to proactively remove outdated cached files without requiring user input. This process effectively frees up internal memory through seamless background services, leading to improved device performance. Moreover, it enhances device security by swiftly and efficiently eliminating potentially sensitive files.

The dynamic cache cleaner assesses two primary factors: device idle time and application usage, logging relevant events. This section will delve into these metrics and how they are analysed to determine when an application's cache should be cleared.

Unlike a gradual process, cache cleaning occurs abruptly. Once the device and an application remain idle for a specified duration, the application's cache is entirely wiped from the device.

Dynamic Catch Cleaning T1, T2 takes 20 minutes interval shows percentage of memory usage by Machine MC1, MC2, MC3, MC4.:

Table-4: Dynamic Cache cleaning memory usage time

	T1	T2	T3		T11	T12	T13
MC1	20	40	49	----	30	25	20
MC2	20	45	55	-----	32	25	22
MC3	18	48	51	-----	35	27	21
MC4	20	45	52	-----	33	30	22
-----	-----	-----	----	----	-----	----	-----

Device Idle Time

There are various approaches to monitoring the idle state of an Android device. This model employs CPU load tracking to gauge device usage. Alternative methods, such as checking the screen status (on or off), impede additional dynamic components, which will be discussed later. By measuring CPU load, the system accommodates minor tasks like checking the time or reading an email or text message, enabling the dynamic cache cleaner to perceive the device as idle during such activities. This behaviour facilitates more efficient cache cleaning. Constantly resetting the service's device idle time tracker due to minor tasks would undermine the dynamic cache cleaner's capability to clear application caches.

Application Usage

Figure illustrates the procedure followed by the dynamic cache cleaner concerning application usage and cache clearance. The concept is straightforward: upon termination of an application with cached

files, the dynamic cache cleaner logs the event. Subsequently, one of two scenarios may occur: either the application is reopened, thereby marking it and preventing its cache from being cleared, or the application's expiration date passes, prompting the dynamic cache cleaner to clear its cache.

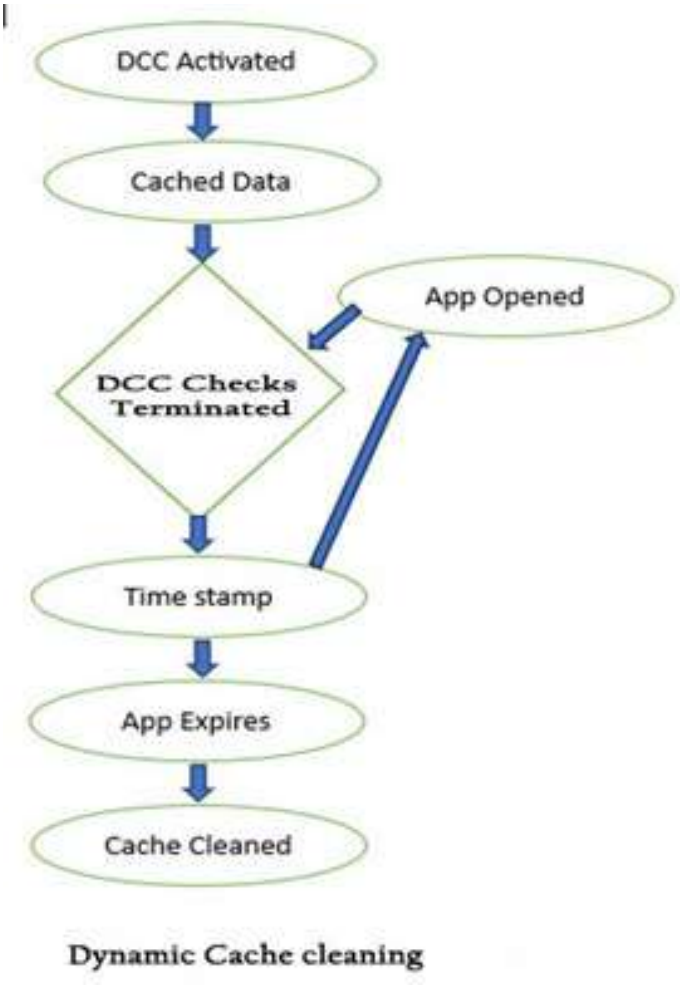


Figure-7: Dynamic Cache cleaning control flow

5.2 Memory Consumption

The dynamic cache cleaner has minimal effect when the device is under heavy use. Under similar conditions, with heavy usage involving all tested applications, the available internal storage after four hours was 2.65GB. This is only 0.29GB less than in the control scenario. This slight difference could possibly be attributed to the dynamic cache cleaner or simply to random actions performed by the user. However, in the control scenario, after setting the device aside and returning to it later, regardless of the duration of absence, the cached storage remains unchanged at 1.26GB. This is not the case when the cleaning service is running. After 24 hours (the default setting) of idle time, application caches are completely cleared from the device.

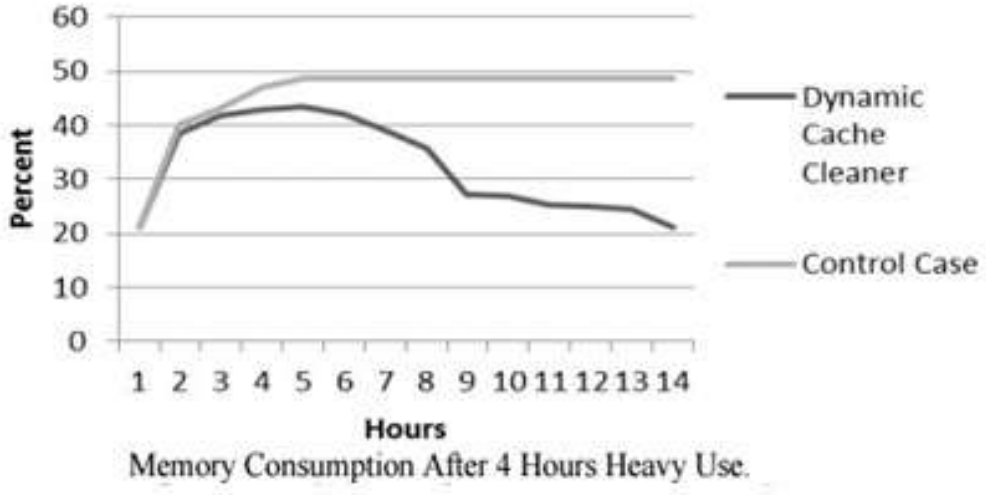


Figure-8

Of particular note is the impact of the dynamic cache cleaner on the device when it's idle. As mentioned earlier, in the control scenario, the device lacks a mechanism to remove stale data within the cache during idle periods. Garbage collection only occurs when the device is actively used and requires additional free internal storage. However, this is not the case with the dynamic cache cleaner. Even during periods of device inactivity, it continues to clear application caches, doing so more effectively.

Similar results were achieved when the device was under heavy use and the dynamic cache cleaner service was running. However, the benefits became evident after the device remained unused following the period of heavy use.

As caches were cleaned, the results began to improve until all application cache memory was cleared, at which point the start time for a single-threaded process returned to 2.28 seconds.

A short user feedback report is listed here. I have shared this application to my students' friends and native circle. After using Dynamic Cache Cleaning application, I request them to send a feedback report through google sheet. The report is listed below.

Table:5 General Feedback of users segregate by age.

	Worst (<40%)	Average (<70%)	Best (>70%)
Male Age 18-35	8.2	76.2	15.6
Female Age 18-35	7.7	77.4	14.9
Male Age 36-50	4.8	71.5	23.7
Female Age 36-50	5.9	75.8	18.3
Male Age Above 50	12.7	67.9	19.4
Female Age Above 50	15,2	73.2	11.6

By observing this feedback report I have found that most of the users made average feedback. In this regard there is a huge scope for further development.

Chapter 6: Methodology Aspects: Device Attachment and Data Security

In a world increasingly dominated by complex technologies, negligent or uninformed use can create opportunities for attack scenarios that might seem improbable to programmers. Modern smartphones, such as those running Apple iOS and Google Android, have the computing power to handle tasks that once required a personal computer. This increased capability has led to the development of numerous applications leveraging new hardware, including internet browsing, email, GPS navigation, personalized messaging, and various other functions. Additionally, mobile devices are an ideal platform for mobile cyber-physical systems, contributing to the substantial amount of personal data stored on these devices.

Among the numerous devices that make up the Internet of Things, smartphones are the most ubiquitous, with many users unaware of the associated risks. This work focuses on security and privacy in mobile systems, specifically targeting a type of attack that exploits USB connection vulnerabilities to access private data on Android devices.

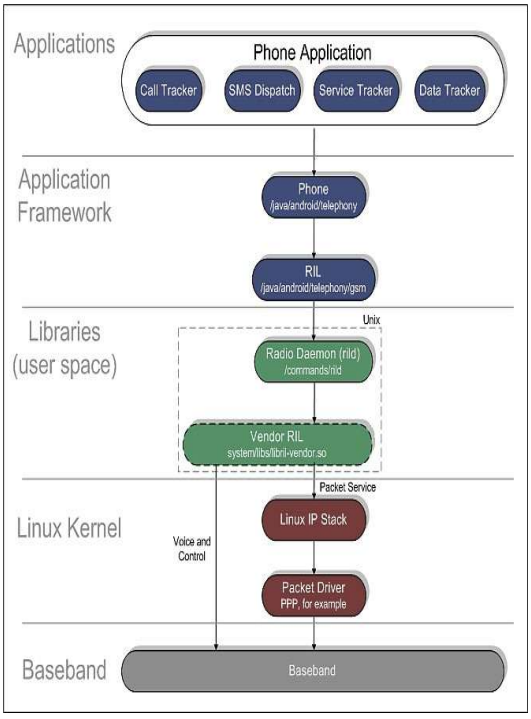


Figure-9: Android Telephony system architecture

The widespread adoption of Android and the common use of Universal Serial Bus (USB) technology have led manufacturers to equip most third-generation smartphones with USB ports. The large user base presents a significant opportunity for malicious attacks: whenever an Android device is connected via a USB port, it is at risk of being compromised. Even minor flaws, errors, or misinterpretations of security specifications can jeopardize system security. Proximity attacks, particularly those exploiting USB connections, illustrate this risk, and anecdotal evidence suggests users often neglect these dangers—a behaviour also observed with USB drives.

6.1 Vulnerabilities in USB Connection

The USB protocol is entirely software controlled, making it uniform across all Android devices. Android supports two sub-protocols: (1) the mass-storage class (e.g., a USB removable disk) and (2) the Android Debug Bridge (ADB). Each peripheral function has an associated device class document specifying the default protocol for that function, allowing hosts and peripherals to interact without detailed knowledge of each other's operations. This class compliance can be risky if the host and peripheral are from different sources.

Initially, only the device's SD card is exposed via USB, not its system or data partitions. When "USB debugging" (ADB) is enabled, the device can be controlled using the "adb" tool provided in the Android SDK. This tool allows inserting and retrieving files, installing APK files, TCP and UDP redirects, and more.

Due to the vulnerabilities associated with this process, Google has implemented a security process for this connection called ADB pairing. This involves pairing the USB connection with an Android Debug Bridge (ADB) using RSA keys accepted by the host computer. Each time the Android device tries to connect to a new host, it must be previously accepted by it.

The Android Debugger Bridge

The Android Debug Bridge (ADB) (Android Open Source Project, 2017) is a command line tool enabling communication with a connected Android device or emulator, allowing various actions through a Unix shell. It facilitates multiple device operations, including app installation and debugging applications. ADB operates as a client-server program with three components: (1) A client, which sends commands and runs on the development computer. The client can be invoked from a command line terminal using an adb command; (2) A daemon, which executes commands on the device and runs as a background process on each emulator or device instance; (3) A server, which manages communication between the client and the daemon, running as a background process on the development computer.

Figure-10: Android debug bridge usage

Once the server configures connections for all emulator instances, adb commands can be used to access those instances. Since the server manages connections and commands from multiple adb clients, any emulator/device instance of any client (or script) can be controlled. More information about ADB can be found on the official site, which is the source of this information (Android Open Source Project, 2017).

6.2. SCENARIOS

Three scenarios for retrieving private information were identified, all associated with USB connections. A script was then implemented to perform the following actions:

1. Obtain device information
2. List all installed packages
3. Make a full copy of the entire SD card contents
4. Copy a file to the device
5. Install an application on the device
6. Run an application on the device
7. Obtain the contact list
8. Obtain the messages
9. Unlock the device screen
10. Overcome ADB pairing
11. Root the device

Three devices, each with a different version of Android, were tested. Specific conditions were also tested for each version. USB debugging is essential to exploit the vulnerabilities considered in this work, as the only contact between the phone and the machine containing the script is through a USB cable.

Attack Scenario I: Configuration

- Hardware: Vivo V2312
- Android version: Funtouch OS14
- USB debugging on
- Device rooted

Achieved results:

1. Obtain device information
2. List all installed packages
3. Make a full copy of the entire SD card contents
4. Copy a file to the device
5. Install an application on the device
6. Run an application on the device
7. Obtain the contact list
8. Obtain the messages

This is the simplest attack scenario, as USB debugging is on and the device is rooted. This device has a specific feature that can increase the vulnerability of USB debugging: the user can "enable" the connection but keep it "inactive," potentially giving a false sense of protection while leaving the door open to an attack. With rooting access, it was possible to obtain all desired information, compromising device security.

Attack Scenario II: Configuration

- Hardware: OPPO CPH1937

- Android version: 11.1
- USB debugging on
- Device "unrooted"

Achieved results:

1. Obtain device information
2. List all installed packages
3. Make a full copy of the entire SD card contents
4. Copy a file to the device
5. Install an application on the device
6. Run an application on the device

If the application roots the device, then the following are also possible: a. Obtain the contact list b. Obtain the messages c. Unlock the device screen

In this scenario, there is no rooting access to the device, which limits the range of possible attacks. However, enabling USB debugging still exposes the device to SD card attacks, specifically the extraction of private data.

Attack Scenario III: Configuration

- Hardware: Samsung SM-M307F/DS
- Android version: 11
- USB debugging off
- Device "unrooted"

Achieved results: It was not possible to extract data from the device. However, it was confirmed that if USB debugging is enabled and ADB pairing is completed, the script can still extract data.

6.3 . IMPLEMENTATION

The developed script is based on the automatic detection of a USB connection: when a device is connected to a computer, it is automatically detected, and the script is initiated. Once the device is identified, various attempts to obtain information are made through Windows PowerShell ISE 5.0 and/or Android Debug Bridge (ADB) commands. All these processes are invisible to the victim and allow access to the entire contents of the SD card and various device information. Figure 11 shows the flow chart of ADB connection process which I have developed for better and secure device attachments process.

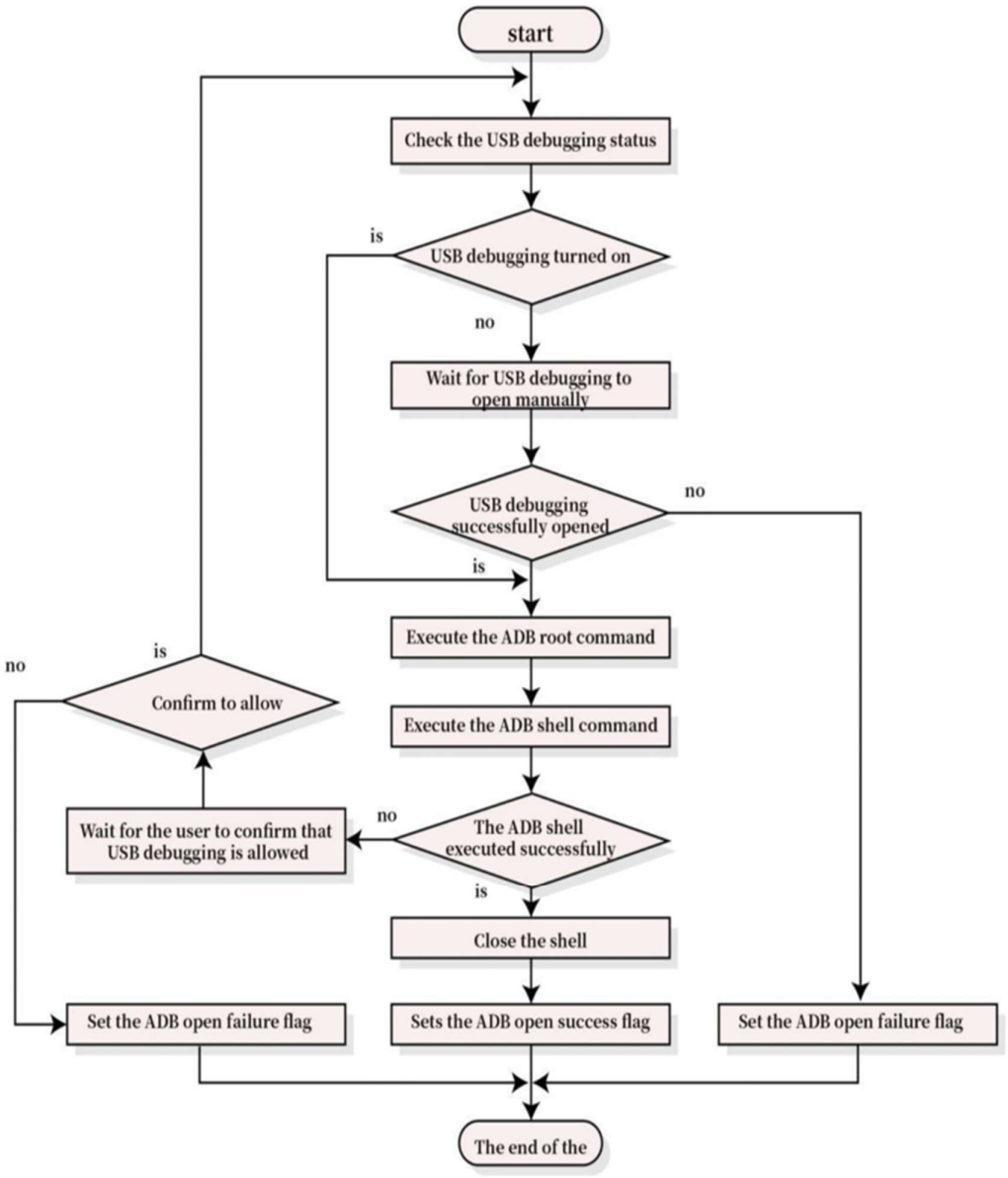


Figure11: ADB Flow Chart

The script performs the following actions, with corresponding results:

1. **Obtain Device Information**: The script retrieves the device's identification, including information such as the letter assigned by the Operating System, the Android version, and the model. This data includes paths needed to execute intended attacks.
2. **List All Installed Packages**: The script obtains a list of all packages installed on the device, used to get the exact name the system assigns to each package. This information is useful for retrieving data about the application installed in order to execute it. The usefulness of this information is explained under "Run an application on the device."
3. **Full Copy of the Entire SD Card Contents**: Since devices have limited storage space, most users save a large amount of personal content on the SD card. This action retrieves all files and folders on the card, such as photographs and movies.

4. **Copy a File to the Device**: After obtaining the SD card contents, the script can copy a file (e.g., an application that allows Android rooting) to the device and verify its presence on the card. This process allows any type of file to be placed on the SD card, potentially compromising the entire device if the file is malicious.
5. **Install an Application on the Device**: The copied file, if it is an application, can be installed on the device. This process is extremely dangerous if the application is malware, as it can corrupt the device in various ways.
6. **Run an Application on the Device**: After installing the application, the script can execute it. If the application is malicious, this raises serious security concerns. This process is facilitated by obtaining a list of all installed applications and their information, enabling the exact path to execute the application to be determined.
7. **Obtain the Contact List**: One significant result is the retrieval of private information from the device, such as the contact list. This is possible if the device is rooted. Since the script can copy, install, and run applications with USB debugging enabled, it can invisibly retrieve this type of information. After granting root access, the script can change access permissions to protected content, allowing the command to copy the information, and retrieve the contact list in SQL and plain text formats.
8. **Obtain the Messages**: Another type of retrieved data is the content of all messages on the device. Using the same process as for the contact list, the script can obtain the message information in SQL and plain text formats.
9. **Unlock the Device Screen**: In the identified scenarios, it is possible to bypass/disable the pattern unlock on Android via ADB commands, but only if two conditions are met: (1) the device is rooted, and (2) the PIN code is known. This is necessary to remove or update the system file containing the screen lock key, which requires changing access permissions and restarting the device.
10. **Overcoming ADB Pairing**: If the device is rooted, the script can remove the system file containing the ADB pairing key, eliminating this security enhancement. However, initial access to the device is required.
11. **Rooting the Device**: The script can root a device by copying, installing, and running an application.

Chapter 7: Conclusion

In this synopsis, I have concentrated on the modeling of the secret key cryptographic algorithm AES, DES, 3DES in the Object-Oriented Programming paradigm. I also use Bayes theorem in machine learning approach, ADB in device connectivity model. The Proposed Object-Oriented model uses the data hiding feature using the secret key. The reuse of code is made possible by implementing the inheritance feature of objects. Generally, all secret key cryptographic algorithms use one secret key in the sender end as well as the receiver end. This secret key can be easily defined in OOP. In my proposed object-oriented model of AES algorithm, different objects of sender class are defined where the secret key of sender class object cannot be known to any third party objects but should be known to any objects of receiver class. I have used AES & 3DES combinedly to protect unauthorized access to our smart phone. AES is faster than DES and DES is provide high security by combining this two provide fast and high security.

My proposed design brings numerous advantages to both software designers and users. It addresses various challenges encountered in software development and enhances software quality. The adoption of object-oriented technology promises increased programmer efficiency, superior software quality, and reduced maintenance costs.

In this framework, ~~we propose a two-tier security~~ A two tier security approach has been proposed incorporating the pipelining of the SIGMA LFSR model. Through the use of inheritance, redundant code can be eliminated, and the functionality of existing classes such as Sender and Receiver can be extended. This enables the construction of programs using standardized modules, resulting in time savings during development and enhanced productivity.

Object-oriented systems are highly scalable, allowing seamless transition from small-scale to large-scale systems. For instance, in our current research model, additional secret keys can be easily incorporated into the Sender class to enable communication with multiple recipients, eliminating the need to rewrite code.

REFERENCES

1. S. Kumar and P. K. Singh, "An overview of modern cache memory and performance analysis of replacement policies," in Proc. of the IEEE International Conference on Engineering and Technology (ICETECH), 2016, Coimbatore, India, pp. 210-214.
2. Y. A. Divya, "An Efficient Virtual Memory using Graceful Code," International Journal of Trend in Scientific Research and Development vol. 3 no. 4, pp. 623-626, 2019.
3. V. Chaplot, "Virtual Memory Benefits and Uses," International Journal of Advance Research in Computer Science and Management Studies, vol. 4, no. 9, 2016.
4. Y. Kim and Y. H. Song, "Impact of processor cache memory on storage performance," in Proc., of the International SoC Design Conference (ISOCC), 2017, Seoul,
5. K. Singh and S. Khanna, "Split Memory Based Memory Architecture with Single-ended High Speed Sensing Circuit to Improve Cache Memory Performance," in Proc. of the 6th International Conference on Signal Processing and Communication (ICSC), 2020, Noida, India,
6. M. T. Banday and M. Khan, "A study of recent advances in cache memories," in Proc. of the International Conference on Contemporary Computing and Informatics (IC3I), 2014, Mysore, India,.
7. Datta, Atanu and Gupta, Somsubhra, Proposed Safety and Security Model for Hand-Held Mobile Devices (March 18, 2022). Available at SRN: https://ssrn.com/abstract=4060930
8. Loreen M. Powell, Jessica Swartz, Michalina Hendon, Awareness of mobile device security and data privacy tools, Issues in Information Systems, Volume 22, Issue 1, 2021 pp. 1-9,
9. Md. Shoriful Islam, Systematic Literature Review: Security Challenges of Mobile Banking and Payments System, Vol. 7, No. 6 (2014), pp. 107-116
10. Trozze A, Kamps J, Akartuna EA, Hetzel FJ, Kleinberg B, Davies T, Johnson SD, Cryptocurrencies and future financial crime. Epub 2022 Jan 5.
11. Weinberg C.B., Otten C., Orbach B., McKenzie J., Gil R., Chisholm D.C., Basuroy S. Technological change and managerial challenges in the movie theater industry. J. Cult. Econ. 2021
12. Mamatzhonovich O.D., Khamidovich O.M., Esonali o'g'li M.Y. Digital Economy: Essence, Features and Stages of Development. Acad. Globe Inderscience Res. 2022 ;3:355–359.

13. Behrouz A. Forouzan, Cryptography and Network Security Special Indian Edition, pp.- 191-224
14. Dr. SunilKarforma, Dr. Sripati Mukhopadhyay, Siddartha Sen, An object oriented approach of Elgamal Digital Signature Algorithm , EAIT 2006 Kolkata. pp.-259,260
15. Liza M. Lowery, Executive Director Department of Telecommunications & Information Services, San Francisco, Developing a Successful E-Government Strategy
16. Alan G. Konheim, Computer Security and Cryptography , Wiley-Interscience A John Wiley & sons, Inc. Publication
17. Russell, D. and Gangemi, Computer Security Basics, G.T, O' Reilly & Associate ,wikipedia.org/wiki/ Advanced_EncryptionSt
18. C Kaufman, R Perlman, M Speciner,Network Security - PHI, Second Edition,2002
19. Collision-resistance of non-uniformly distributed functions. In Tsuyoshi Takagi, editor, *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016*, pages 79–85. Springer, Heidelberg, 2016.
20. [UMHA16] Rei Ueno, Sumio Morioka, Naofumi Homma, and Takafumi Aoki. A high throughput/gate AES hardware architecture by compressing encryption and decryption datapaths - toward efficient CBC-mode implementation. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *CHES 2016*, volume 9813 of *LNCS*, pages 538–558. Springer, Heidelberg, August 2016.
21. Prabhdeep Singh and Ashish Kumar Pandey, "A Review on Cloud Data Security Challenges and existing Countermeasures in Cloud Computing," Int. J. Data Informatics Intell. Comput., vol. 1, no. 2, pp. 23–33, Dec. 2022, doi: 10.59461/ijdiic.v1i2.33.
22. N. Rajesh and A. A. L. Selvakumar, "Hiding personalised anonymity of attributes using privacy preserving data mining," Int. J. Adv. Intell. Paradig., vol. 7, no. 3/4, p. 394, 2015, doi: 10.1504/IJAIP.2015.073717.
23. W. Wang, L. Ying, and J. Zhang, "On the Relation Between Identifiability, Differential Privacy, and MutualInformation Privacy," IEEE Trans. Inf. Theory, vol. 62, no. 9, pp. 5018–5029, Sep. 2016, doi: 10.1109/TIT.2016.2584610.
24. J. Pei, K. Zhong, M. A. Jan, and J. Li, "RETRACTED: Personalized federated learning framework for network traffic anomaly detection," Comput. Networks, vol. 209, p. 108906, May 2022, doi: 10.1016/j.comnet.2022.108906.
25. G. K. Ragesh and A. Kumar, "Trust-based secure routing and message delivery protocol for signal processing attacks in IoT applications," J. Supercomput., vol. 79, no. 3, pp. 2882–2909, Feb. 2023, doi: 10.1007/s11227-022-04766-z.
26. A. Al Hayajneh, M. Z. A. Bhuiyan, and I. McAndrew, "Improving Internet of Things (IoT) Security with Software-Defined Networking (SDN)," Computers, vol. 9, no. 1, p. 8, Feb. 2020, doi: 10.3390/computers9010008.
27. B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," J. Netw. Comput. Appl., vol. 84, pp. 25–37, Apr. 2017, doi: 10.1016/j.jnca.2017.02.009.
28. J. Qiu, Z. Tian, C. Du, Q. Zuo, S. Su, and B. Fang, "A Survey on Access Control in the Age of Internet of Things," IEEE Internet Things J., vol. 7, no. 6, pp. 4682–4696, Jun. 2020, doi: 10.1109/JIOT.2020.2969326.
29. Dr. S. Smys, Dr. Abul Basar, and Dr. Haoxiang Wang, "Hybrid Intrusion Detection System for Internet of Things (IoT)," J. ISMAC, vol. 2, no. 4, pp. 190–199, Sep. 2020, doi: 10.36548/jismac.2020.4.002.
30. T. Saba, A. Rehman, T. Sadad, H. Kolivand, and S. A. Bahaj, "Anomaly-based intrusion detection system for IoT networks through deep learning model," Comput. Electr. Eng., vol. 99, p. 107810, Apr. 2022, doi: 10.1016/j.compeleceng.2022.107810.
31. A. Fatani, A. Dahou, M. A. A. Al-qaness, S. Lu, and M. A. Abd Elaziz, "Advanced Feature Extraction and Selection Approach Using Deep Learning and Aquila Optimizer for IoT Intrusion Detection System," Sensors, vol. 22, no. 1, p. 140, Dec. 2021, doi: 10.3390/s22010140.

List of Publication

[1] Datta, Atanu and Gupta, Somsubhra, "Proposed Safety and Security Model for Hand-Held Mobile Devices", Social Science Research Network Electronic Journal, ISSN: 1556-5068, SSRN, **ELSEVIER** , March 18, 2022, Available at SSRN: https://ssrn.com/abstract=4060930 or http://dx.doi.org/10.2139/ssrn.4060930

[2] A Datta, S Gupta, "Ubiquitous Devices Safety using Cyber Security", ICIS 2022, published as Research Article, AIP proceedings ISSN:1551-7616, **ISBN:** 978-0-7354-475, Vol: 2878, 020014-1–020014-5, 2023. (https://doi.org/10.1063/5.0171358) **(AIP proceedings; both Scopus and Web of Science indexed)**

[3] D. Ghosh, K. Ghosh, C. Chakraborty, A. Datta, S. Gupta, "Securing the Future: Emerging Threats and Countermeasures in Cryptography", Securing the Digital Frontier, K. Sharma Eds. Scrivener Publishing LLC, Chapter 5, pp. 91-108, 2025

[4] A. Datta, S. Gupta, S. Som, "Addressing security challenges of Payment Systems in hand-held devices", accepted for publication in Book Chapter, *Recent Advancements in Computational Intelligence and Design Engineering* Rouledge **CRC Press, Taylor & Francis**, ISBN: 978103298036, February 2025. **(In Press: Scopus indexed)**

[5] A. Datta, S. Gupta, S. Som, "Exploring USB security of Hand-held Devices", published as Book chapter at **Fusion of Minds: Multidisciplinary Research in Emerging Technologies** (Dr.R. Kumar Eds.), ISBN 978- 93-6135-962-0, DOI: https://doi.org/10.22271/ed.book.2972 , Publication date: 7th oct 2024, through International Conference on Integrative Science and Engineering (ICISE) 2024.

[6] A. Datta, S.Gupta, S.Som, "Ubiquitous Hand held device's SIGMA LFSR driven security model", Educational Administration Theory and Practise, ISSN 2148-2403, Vol. 30, No: 1, pp. 4488 - 4492, January 2025 (**SCOPUS Indexed**)

[7] A. Datta, S.Gupta, S.Som,"Sigma LFSR based security framework for fees payment transaction online through hand held devices", Paper ID: 60, Accepted for presentation at Scopus Conference RTET 2024, Hinweis Research, 2024. **(Scopus indexed)**